Amendments to the Claims

1.   (Currently Amended) A computerized method of creating test coverage for non-deterministic programs <u>within a testing environment</u> comprising:

receiving a graph of edges and states representing a program under test;

creating a continuous cycle of edges through the graph that reaches each edge in the graph at least once;

splitting the continuous cycle into discrete sequences that end at edges reaching non-deterministic states <u>uncontrollable by the testing environment</u> in the graph;

executing the program <u>under test</u>and verifying that the program conforms to the behavior represented by discrete sequences;

determining untested program behavior as discrete sequences not reached by the program;

creating strategies through the graph that have a higher probability of reaching discrete sequences not reached by the program;

<u>storing a representation of the created strategies in computer memory;</u> and

executing the program <u>under test</u> under test conditions <u>using the stored created strategies</u> that cause the program to <u>have a higher probability to</u> execute through states that correspond to the created strategies <u>untested program behavior</u>.

'·2.   (Original) The method of claim 1 wherein the received graph is a set of states and a set of edges, and edges are represented as state source-target pairs.

3.   (Original) The method of claim 1 wherein the continuous cycle of edges is created from the graph input using a Chinese Postman tour algorithm.

4.   (Original) The method of claim 1 wherein the graph states are received as a set of deterministic vertices and a set of non-deterministic vertices.

5.   (Currently Amended) The method of claim 1 wherein the executing program is
instrumented with executable code that verifies ~~upon~~ during execution that a program state
conforms to a state of the graph.

6.   (Original) The method of claim 1 wherein created strategies are inputs that
represent edges between states of the graph, and test conditions cause the program to enter
untested program behavior.

7.   (Original) A computer system comprising:
memory and a central processing unit executing,
        a compiler for compiling an executable specification into an abstract state
machine,
        a graphing program for creating a continuous cycle touching all edges of the
abstract state machine, and for splitting the continuous cycle into discrete sequences that
end at non-deterministic states;
        a strategy calculation program for creating strategies more likely to reach the
untouched discrete sequences;
        a coverage program for executing a program and verifying that the program
executes states corresponding to those modeled by discrete sequences of the abstract state
machine and for determining untouched discrete sequences and for executing the
program according to the created strategies and verifying whether the program executes
states corresponding to the untouched discrete sequences.

8.   (Original) The system of claim 7 wherein a continuous cycle is determined
according to a Chinese Postman algorithm.

9.   (Original) The system of claim 7 wherein discrete sequences comprise beginning
states reachable from edges exiting non-deterministic states.

10.   (Original) The system of claim 7 wherein an untouched discrete sequence is a
state selectable from a program code executing at a remote computer.

11.   (Original) The system of claim 7 wherein the abstract state machine comprises a graph of states and edges.

12.   (Original) The system of claim 11 wherein the strategy calculation program receives the graph and an edge probability function as input.

13.   (Original) The system of claim 7 wherein untouched discrete sequences represent less than 10% of the discrete sequences and all untouched discrete sequences are touched when the program is executed according to the created strategies.

14.   (Original) The system of claim 7 wherein not all untouched discrete sequences are verified when the program is executed according to the created strategies.

15.   (Currently Amended) A computer-readable medium having thereon computer-executable instructions comprising:

instructions for creating a model of modeling program behavior comprising an abstract state machine with edge transitions;

instructions for verifying program behavior;

instructions for splitting a the model of program behavior into sequences of at least two edge transitions ending at non-deterministic behavior;

instructions for determining strategies for the sequences more likely to reach an identified program behavior; and

instructions for causing a program to execute behavior corresponding to strategies for the sequences more likely to reach the identified program behavior.

16.   (Original) The computer-readable medium of claim 15 wherein the modeled program behavior is modeled as an abstract state machine.

17.   (Canceled)

18.   (Original) The computer-readable medium of claim 15 wherein the non-deterministic behavior comprises communications with a remote computer.

19.   (Original) The computer-readable medium of claim 15 wherein determined strategies are determined based on a comparison of edges exiting a deterministic state representing program behavior, and a selected edge has a highest probability of reaching a state representing the identified program behavior.

20.   (Original) The computer-readable medium of claim 15 wherein the instructions for verifying program behavior cause the program to execute code that verifies that the program is in an expected model state.